INTERIM REPORT

G52GRP YEAR 2 GROUP PROJECT

Group 2: Lim Zhi En (khcy2lzn), Tang Wei Qi (khcy2twi), Michael Muita Mugo (kecy2mmg), Mohammad Syamil(khcy2msr)

12/9/2013

TABLE OF CONTENTS

INTRODUCTION

1.1 PROJECT BACKGROUND AND RESEARCH

Motor vehicles play a conspicuous role in the modern human life. Transport (especially cars) offers rapid, reliable and convenient mobility on demand to an ever-growing number of people in countries throughout the world. But for all their positives, automobiles carry with them many negatives. One of which is that it's difficult to find car park especially in public place such as colleges and universities. So do students and staff at the University of Nottingham Malaysia Campus. This results in most students being late for classes. This problem became more complicated with the recent changes in the traffic policy. With this problem in mind, there are a lot of solutions implemented to derive from, that help to overcome this problem. We can observe such measures being implemented in public places such as shopping malls.

Tracing along the same line of thought from existing solutions, we decided to develop an android application with the aim of assisting in finding free parking slots easily. Secondary research was carried out about existing systems in the market. This led to the conclusion that the existing system can be improved. We figured out that the existing system is not effective enough, as they just tell the drivers how many empty parking slots are left. This help users a lot but, but is still insufficient because the user does not know where exactly the empty car park is located. They still need to go around the car parks and randomly search where the free parking sis located. This leads to higher fuel consumption than needed.

With this in mind, the system to be developed shall help users locate where the available parking bays are, and to also save its parking location for easier location by the driver.

1.2 PROBLEM DESCRIPTION

The initial brief for the coursework is to develop a wireless parking system for the UNMC. In doing so, the system would ultimately help the students and lecturers with finding parking in close proximity to their destination, with a visual layout of parking availability and to locate where their car is parked. This would be time and cost saving for them. In order to achieve this, it is necessary to access park mapping data, from surveys carried out by the engineering department.

The application shall have the proceeding functions for all members of UNMC. The application shall be able to help the user to find parking locations and if they're free or not. Moreover, this shall make use of old and unused Android mobile phones, by utilizing their cameras to detect the presence of a vehicle. Once the user has parked their vehicle, the application will give the option of saving the vehicle location in the in-built mapping that allows less strenuous searching for forgetful people. The application shall provide real time updates for information and will need constant access to the internet.

On startup, the application will load with a welcome screen. This will then lead to a GUI map with real time data of parking status. The application shall then be continuously refreshing and syncing data about parking availability with a server containing a database of this. The driver shall then use the android application to head towards the parking of their desired location or through the use of a noticeboard to find the section where the parking is located in the car park. The driver then parks the car and the parking status of the particular parking is changed to: occupied; via image processing which is used to detect the presence of the vehicle. This is then synced with the whole wireless parking system and the driver is then given the option to save their vehicle's location.

1.3 CURRENT AND EXISTING SYSTEM

Based on our research on existing market software systems, we discovered and studied a few similar systems that we found. We managed to extract the pros and cons of each system and analyzed them. It gave us a rough idea on what we actually want and need in our own system.

1.3.1 MY CAR PARKING BY DRD

- It's a currently free application developed for android platform by Drd.
- Allows the users to save parking location by utilizing GPS precision signal.
- If GPS signal is not available, user could take a picture to save their position.
- It records parking location and provide guidance back to car location with the help of Google Navigator instruction based on Google Map service.

- Allows the users to set a reminder for parking ticket expiry with a notification alert.[1]

1.3.2 SUNWAY PYRAMID SHOPPING MALL CAR PARK GUIDING SYSTEM

- Red/green indicator to guide drivers on available parking bays.
- Directions indication to where zones that have free parking bays.
- Uses infrared and ultrasound as sensor to detect cars.
- LED lights installed along the driveway to indicate availability where red means occupied and green otherwise.
- There's a few second delay in taking a reading by the sensor.
- Display number of parking bays available in a particular zone on sign boards.
- Real-time detection of parking bays availability. [2]

1.3.3 SENSIT PARKING SPACE OCCUPANCY SYSTEM BY NEDAP AVI

- Display number of parking bays available in a particular zone on sign boards.
- Uses wireless infrared and magnetic sensor to detect vehicle. Dual-sensor technology to ensure more accurate detection.
- Wirelessly data collectors (SENSIT nodes) will collect and send data to an online server for further usage and analysis.
- Real-time detection of parking bays availability. [3]

My Car Parking has good implementation of google map and GPS technology in their system but rather differs in terms of vision and scope from that of our proposed system. My Car Parking only focuses on helping the users return to their cars while our proposed system has a broader functional scope if compared with My Car Parking. On the other hand, the parking system implemented in Sunway Pyramid has similar goals as ours. The only thing it lacks is the absence of mapping tools for the users to easily locate available parking bays. We also came across SENSIT during our research. In

[1] My Car Parking - Android Apps on Google Play. (n.d.). Retrieved December 9, 2013, from https://play.google.com/store/apps/details?id=it.drd.uultimatecarfindparking&hl=en

[2] Shopping Mall Malaysia, Sunway Pyramid. (n.d.). Retrieved December 9, 2013, from http://www.sunway.com.my/pyramid/corpInfo/mallServ_FAQ.asp

[3] Vehicle detection by Nedap AVI. (n.d.). Retrieved December 9, 2013, from http://www.nedapavi.com/solutions/vehicle-detection.html

fact, SENSIT is the closest to what we are trying to achieve. It uses advanced technology in their vehicle detection with its self-developed wireless sensors. Nonetheless, there is no rules that state we cannot reinvent yet another existing system as long as we come up with our genuine implementation. Instead of going with the sophisticated infrared and magnetic sensors, we have a way cheaper and eco-friendly alternative, which for the scope of project we're given as university students, is rather reasonable and appropriate.

1.4 DEVELOPMENT METHODOLOGY

The proposed project that we are going to undertake is a system designed to improve a targeted community, Nottingham Malaysia Campus and overcome certain issues like the usability of the parking system and the overwhelming electronic waste. We decided to adopt the agile methodology as our software development methodology because it is best suited to our scenario. The main reason we chose agile over a conventional process model is that we will deliver better in a working environment which is centered on seamless communication and interaction between team members since we are only a four-man team. With such a small organization, it is always handy to arrange informal meetings to discuss on our progress and face-to-face communication can yield comparatively higher efficiency in our productivity as the team can remain compact and closely-knitted all the time. Agile methodology encourages customer involvement in the software development process. Hence, with the customers as a part of the team, they as the domain expert can help in our requirement analysis significantly and constantly reviewing our progress to keep us on track. We need a process model which is flexible and can easily accommodate changing requirements. Since there are some uncertainty in the requirement and design, with agile methodology, we can better adapt to uncertainty faced during requirement analysis. Since we can iteratively repeat our entire development cycle, we can review our system requirement frequently and make changes if there is a need.

We plan to practice the five values of Extreme Programming on top of Agile Methodology in our system design and implementation. The five values are communication, simplicity, feedback, respect and courage. These values are founded upon the basis of agile methodology. We will prioritize communication with our stakeholders and get valuable feedback from them by delivering frequent small releases. We will also encourage simplicity in design and coding so that refactoring of code will be very much easier whenever there are changes in the requirements. Fundamentally, we respect each other and appreciate the effort by each and every member of the team.

Extreme programming is an iterative development process. In order to facilitate our progress, we will have an informal and a formal meeting per week. A formal meeting is held with our supervisor and stakeholders while an informal meeting is just held among the group members. In each informal meeting we will review on previously assigned tasks and what needs to be done next. We will make sure that each group member has been assigned task at the end of every informal meeting. The purpose of a formal meeting is to report our progress to our supervisor and also seek guidance from him.

1.4.1 PRACTICES AND STEPS TO BE TAKEN IN DEVELOPMENT

ITERATIONS

Unlike conventional waterfall development methodology, our development phases are far more flexible as we continuously iterate between planning and implementation phases. Being more specific, we will go through requirement analysis, software design, coding and testing within every iteration. However, the structure of these phases does not mean that we have to stay rigid and follow the sequence of the phases. We welcome active feedback from our stakeholders and we will not be afraid to implement further changes to our system requirements. Having said so, we therefore are not restricted from jumping back to previous stages if we feel like making some adjustment to it. Minutes of meeting are recorded and documented after every meeting to keep track of progresses and status.

COMMUNICATION

Our group members maintain close communication with each other either through face-to-face or online conversation. This is essential so that each member is easily approachable and there is transparency in everyone’s working progress. Off-school, we make good use of the popular smartphone instant messaging service, Whatsapp to conduct discussion regarding project tasks and also meeting schedule. In this Whatsapp group chat, everyone has an equal opportunity to voice out their personal opinions and help each other out. For task delegation, we use a Web-based application called Asana to assign tasks to every group member immediately after each informal meeting. This provides a guideline and reminders for us to work on our parts as the complexity level rises, so that there will be no excuses for us to not complete our tasks in each sprint cycle.

SIMPLICITY

We exercise simplicity in design, from source code to user interface. Simplicity is an important aspect to consider in our software design because as the system becomes too complex, the dependencies within the system may cease to clear without proper designing. With Java as our primary language, we can make the best out of the object-oriented paradigm by maximizing code reuse and modularity. We will also employ the functionality of the existing libraries to avoid creating something redundant which is already out there available for us.

PAIR PROGRAMMING

Pair programming is definitely the essence of extreme programming methodology. Since solo programming can be very much error prone and less efficient, pair programming will minimize these weaknesses by having an extra brain and pair of eyes doing the same job. We will divide our group into two pairs and we will run our coding tasks in parallel on separate program components or modules. For each pair programmers, one will be doing the coding while another will overlook the entire code blocks and scan for errors and bugs to be fixed. By doing this, not only we will speed up the pace tremendously, the practice will also yield higher quality code.

TESTING

Our application system constitutes of several components working together, such as the sensor, the user application and the server. Thus, we will need to do a lot of testing on these components to make sure that the integration of these components will be successful. In iteration we will conduct unit testing on each individual component to ensure that it functions properly on its own. When all components have passed the unit testing, we will carry out the integration testing to fix any errors that manifest when we put all the components together. User acceptance test is then executed by the stakeholders at the end of each iteration cycle to check whether it meets the customer’s requirements. These repetitive testing and checking with the customers ensures that we are making improvements and moving closer to our goal at the end of each development cycle.

1.5 ORGANIZATIONAL STRUCTURE

Due to our group size, most of us are required to take up multiple roles in the team. Roles are discussed and distributed based on each member's strength. Although each member is allocated with a set of specific responsibilities, every member is encouraged to be flexible and help out with each other duties. As for leader, he shall be the one who overview the project status and making sure procedures are on time and carried out smoothly across the development process. It is also important for the leader to keep every member committed in the project to ensure integrity. The roles are distributed as table follows:

ROLES	MEMBER
Leader	Zhi En
Open Day Producer	Zhi En, Michael
Editor in Chief	Syamil
Artistic Director	Wei Qi
Repository Master	Wei Qi
Software Architecture	Everyone
Quality Control	Everyone
Coding	Zhi En, Wei Qi, Michael

BRIEF DESCRIPTION ON EACH ROLE

LEADER: Oversees the entire group progress and leads the group to the right direction.

OPEN DAY PRODUCER: In charge of the planning and setting up of booth during Open Day.

EDITOR IN CHIEF: Takes care of project documentation and archives project artifacts.

ARTISTIC DIRECTOR: In charge of the aesthetics of our project deliverables.

REPOSITORY MASTER: Organizes the file hierarchy in the GitHub repository.

SOFTWARE ARCHITECTURE: Plans the architectural design of the software system.

QUALITY CONTROL: Conducts software testing and ensures that the software passes all tests.

CODING: Constructs the software from the source code level.

MEETING ORGANIZERS

There will be a chairperson and a secretary for each meeting. These two roles will rotate between the group members on a weekly basis. The purpose of setting the roles on rotation is so that everyone can participate equally in the meeting procedures.

SYSTEM SPECIFICATION

2.1 USER STORIES

As a	I want to	So that
Student	Find parking close to my classes	I don’t have to walk for a long distance
All Users	Have more efficient parking system	I could spent less time looking for car park
Lecturer	Park close to my office	Less time is needed to travel there
All Users	Find parking spaces quickly	Less petrol is used and less pollution done
All Users	Quickly get to my car	I can go home sooner

All Users	Know location of available parking lots	I don’t waste time looking for one
All Users	Have an easy to use system	Don’t need to learn many things to use it
All Users	Be reminded where I parked my vehicle	I can remember where I parked it
All Users	Check real-time free parking lots	I won’t waste time heading towards a taken parking space
All Users	Know how many parking spaces are there in various zones	I have a rough idea where to head to
Developer	Have a vehicle detection algorithm by image processing	Detect availability of a parking lot from image taken
Developer	Have an old android phone	I could use its camera to detect parking lot availability
Developer	Have a server	I have a communication platform between user application and sensors
Developer	Have a wide compatible interface	Most user may be able to use developed application
Developer	Have a counter display	I could inform users how many parking is available at various zones
Developer	Have a secure server	Data integrity could be maintained
Developer	Have power supply to sensor phone	It could operate for long hours
Developer	Have an online database	Could store data in an organized way
Developer	Have a shelter for sensor phones	It won’t be damaged by weather

2.2 FUNCTIONAL REQUIREMENTS

- The mobile application should be able to display the university car park layout.
  - This will need to be split up in various zones, according to various departments and major buildings.
  - This will also need to be scaled to enable for zooming in and out.

- The system should be able to differentiate between red and yellow zone parking.
  - Different color label is needed.

- The system shall need to very frequently update the status of parking availability.
  - The system should grey out parking spots once they are taken.
  - The system should also highlight parking spots immediately they are available.
  - Synchronization is required to be often between various system components, with minimal downtime.

- The system should enable users to zoom into specific zones for more detailed information regarding parking.
  - This zooming will need to be enabled in the mobile application, through touch that will zoom in once a user selects a particular parking zone, within the full map.
  - Upon zooming in, the complete layout of that particular zone should be shown.
  - The user should be able to zoom out of the full map.

- The system needs to have a parking counter.
  - The counter should be updated in real-time.
  - There will be a counter that will show the number of parking slots available in the entire campus while in the full map mode and counters for other specific parking zones.

- The system should be enabled to query buildings or location names.
    - The system will have a database of university buildings, locations and landmarks.
    - Querying shall be performed with the addition of a search bar in the mobile application.
    - Querying details will include options for red or yellow zone parking and location name.
    - If the user's query matches what is in the database, the system shall automatically search for parking zone nearest to the queried location which has available parking spaces of the specified type.
    - Upon successful search, the system will zoom into that particular parking zone map to notify the user of its location.

- The system should remember where the user is parked.
    - The system should allow a user to mark their parking location on the map, so that it can be returned to in case they forget where they are parked.

- The system should be user friendly.
    - The mobile application should have intuitive and trendy navigation.
    - The mobile application should have a neat and simple interface.
    - The system should run smoothly and load quickly.
    -

2.3 NON-FUNCTIONAL REQUIREMENTS

- Usability
    - The mobile application should be usable by people of all ages, gender, races and cultures and also education background as long as they are vehicle owners and are in need of parking service in campus.
    - The mobile application should have a simple and intuitive user interface so that even new users can easily learn how to use it in a few runs.

- The users will need Internet access on their mobile phones in order to activate the mobile application and receive real-time update on the car park status.
- The mobile application is only supported on Android platform.

- Performance
  - The system will need to have a vehicle detection algorithm that is optimized enough to be able to run on low-end android phones.
  - The system will need to have a centralized server and database to store the data to be shared among clients.
  - The system will need to be mainly coded and supported in Java, due to the nature of Android, which is the targeted platform. The database backend will be programmed in SQL language.
  - The sensors will need to communicate with the server via a wireless network. Therefore, parking lots will need to be equipped with Wi-Fi coverage with internet connection.
  - Due to the GPS function not included in this system, users are not able to track their current location on the map.

- Dependability
  - Android phones with built-in camera to be utilized as vehicle sensors.
  - A counter display, this can be an LCD monitor as we do not possess the necessary knowledge to construct an LED display board.
  - The Android phones (sensors) will need to have sufficient power supply to minimize the occurrence of shutdowns in the middle of operations.
  - A shelter for the Android phone is required so that they will be less susceptible to various external damaging factors.

- Security
  - The server shall be secured so that the information kept and transmitted across components will not be intercepted and sabotaged to ensure a smooth flow of operations.

- Operating constraints

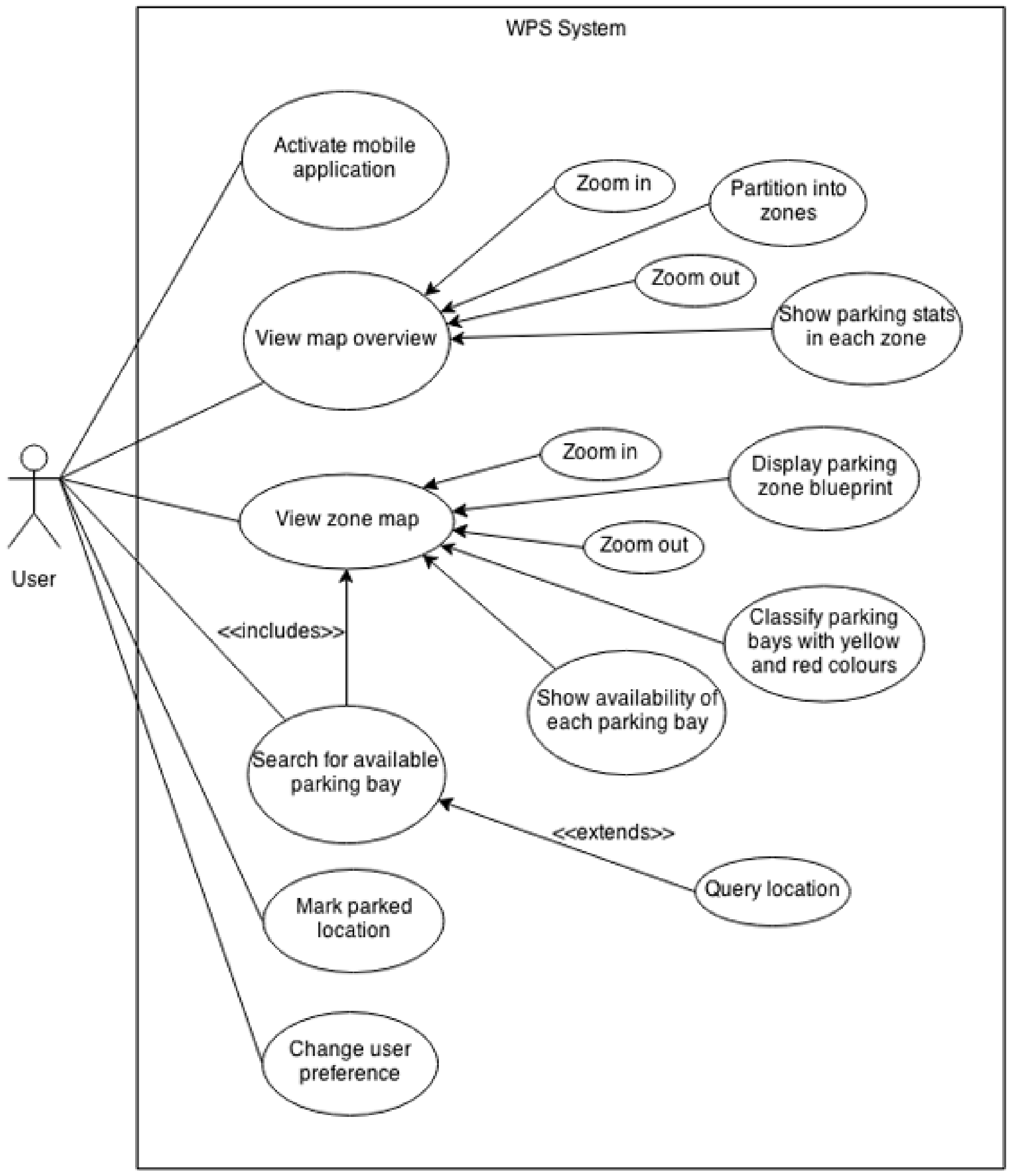
- We are supposed to deliver our entire project, including documentation and software, in two-semester time.
- As we are not provided any funds by the university, we will need to get hold on unused old Android phones to utilize it as the vehicle sensors.

2.4 ASSUMPTIONS AND DEPENDENCIES

- The lifespan of the mobile phone will be shortened due to various external factors such as harsh weather, exposure to sunlight and heat, vandalism, accidents etc.
- Developed system will be used by the staffs and students of Nottingham Malaysia Campus.
- System status has to be updated frequent enough so that information won't be outdated.
- The user will drive carefully while searching for parking spaces and browsing through the developed mobile app at the same time.
- The user interface of the developed mobile app has to be user friendly and understood in order to locate the available parking spaces efficiently.
- The LCD car park counter display should display the correct number of empty parking spaces in each parking zones.

DESIGN SPECIFICATIONS

3.1 USE CASES



WPS System
Activate mobile application
Zoom in
Partition into zones
Zoom out
Show parking stats in each zone
View map overview
Zoom in
Display parking zone blueprint
View zone map
Zoom out
User
<<includes>>
Classify parking bays with yellow and red colours
Show availability of each parking bay
Search for available parking bay
<<extends>>
Query location
Mark parked location
Change user preference

Figure 3.1.1: Use Case Diagram 1

Use Case Name: Activate mobile application	ID Number: 1
Short Description: This describes how to start the app.	
Primary Actor: User	
Trigger: User selects the app icon in the android application menu.	
Main Success Scenario: 1. User goes to Android application menu. 2. User looks for app icon. 3. User touches on app icon.	
Extensions: 1. User's phone is not turned on. User switches on the phone. 2. App is not installed. User downloads and installs the app from Google Play.	

Use Case Name: View map overview	ID Number: 2
Short Description: This describes how to view the UNMC car park layout.	
Primary Actor: User	
Trigger: User selects the *Map* button.	
Main Success Scenario: 1. User touches the *Map* button on the app's main menu. 2. The app screen displays the full car park layout map of UNMC, which is divided into several parking zones. 3. Floating notification icon on each parking zone shows the number of parking bays left. 4. User zooms in or zooms out to have a finer view of the map.	
Extensions: 1. ALTERNATIVE: 1. Use Case 7: Launch sidebar	

Use Case Name: View zone map	ID Number: 3
Short Description: This describes how to view the detailed map of each parking zone.	
Primary Actor: User	
Trigger: User touches parking zone screen area.	
Main Success Scenario: 1. User touches a particular parking zone screen area on the full map. 2. User enters the full map overview screen. 3. User touches on the parking zone area (e.g. Zone A) which he wants to view in details. 4. The app screen displays the Zone A car park layout with each and every parking bay on its respective location. 5. User distinguishes the two types of parking bays by recognizing their colour labels, which are red and yellow. 6. User recognizes that a parking bay is occupied at the moment if it is greyed out on the map. 7. User zooms in or zooms out to have a finer view of the map.	
Extensions: 1. ALTERNATIVE: Use Case 7: Launch Sidebar 2. ALTERNATIVE: User selects any of the tabs on the tab bar to navigate to different parking zone maps.	

Use Case Name: Query location	ID Number: 4
Short Description: This describes how to find parking bay nearest to the queried location.	
Primary Actor: User	

Trigger: User selects the *Find* button.
Main Success Scenario: 1. User touches the *Find* button on the app's main menu. 2. User enters the *Find* Parking interface. 3. In the Include option, user selects the type of parking bay to be allowed for the query. 4. In the Nearest to option, user selects his destination from the drop-down menu. 5. The display area shows the nearest zone map in relative to the queried location which has empty parking bays.
Extensions: 1. ALTERNATIVE: Use Case 7: Launch Sidebar 3. User cannot find his destination from the drop-down menu. User finds available parking bay by manually navigating through the map.

Use Case Name: Mark parked location	ID Number: 5
Short Description: This describes how to make the app remembers where the user parked.	
Primary Actor: User	
Trigger: User holds his touch on the parking bay screen area for 3 seconds.	
Main Success Scenario: 1. A *You Are Here* pin appears above the specified parking bay. 2. User retrieves his last parked location in the *Where* interface. 3. User clears the memory by touching the *Reset* button.	
Extensions:	

Use Case Name: Change user preference	ID Number: 6
Short Description: This describes how to change the configuration of the app to suit personal preference.	
Primary Actor: User	
Trigger: User selects the *Settings* button.	
Main Success Scenario: 1. User touches the *Settings* button on the app's main menu. 2. User enters the *Settings* interface. 3. User toggles on or off the switches corresponding to various aspects of the system environment.	
Extensions: 1. ALTERNATIVE Use Case 7: Launch Sidebar	

Use Case Name: Launch sidebar	ID Number: 7
Short Description: This describes how to brings out the sidebar of the app.	
Primary Actor: User	
Trigger: User swipes from left edge to right.	
Main Success Scenario: 2. User swipes his finger from the left edge to the right of the screen. 3. The app's sidebar appears on the screen. 4. User selects which interface he wants to navigate to.	
Extensions: 1. ALTERNATIVE User touches the button with three horizontal lines which is located on the top-left corner of the screen.	

Use Case Name: Hide sidebar	ID Number: 8
Short Description: This describes how to hide the sidebar from the screen.	
Primary Actor: User	
Trigger: User swipes from right to left.	
Main Success Scenario: 1. User swipes his finger from the right to the left of the screen. 2. The app's sidebar slides off the screen.	
Extensions: 1. ALTERNATIVE User touches the button with three horizontal lines which is located on the top-left corner of the screen.	

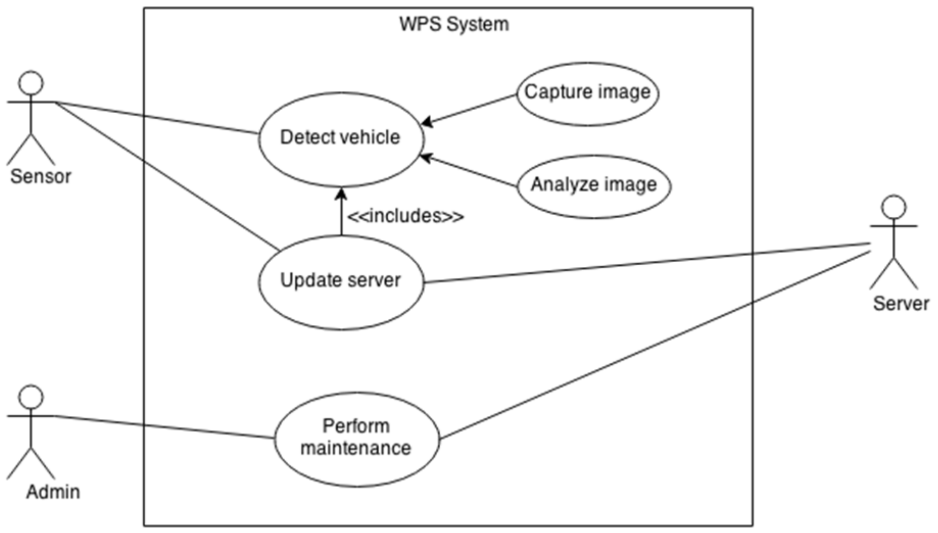


Figure 3.1.2: Use Case Diagram 2

Use Case Name: Detect vehicle	ID Number: 9
Short Description: This describes how to detect whether a parking bay is occupied or not.	
Primary Actor: Sensor	
Trigger: Auto	
Main Success Scenario: 1. Sensor camera takes photo of the parking bay. 2. Sensor analyses the photo using image processing algorithm. 3. Sensor transmits the result of the image processing to the server. 4. Server updates its database using the received data.	
Extensions:	

3.2 CONTEXT DIAGRAM

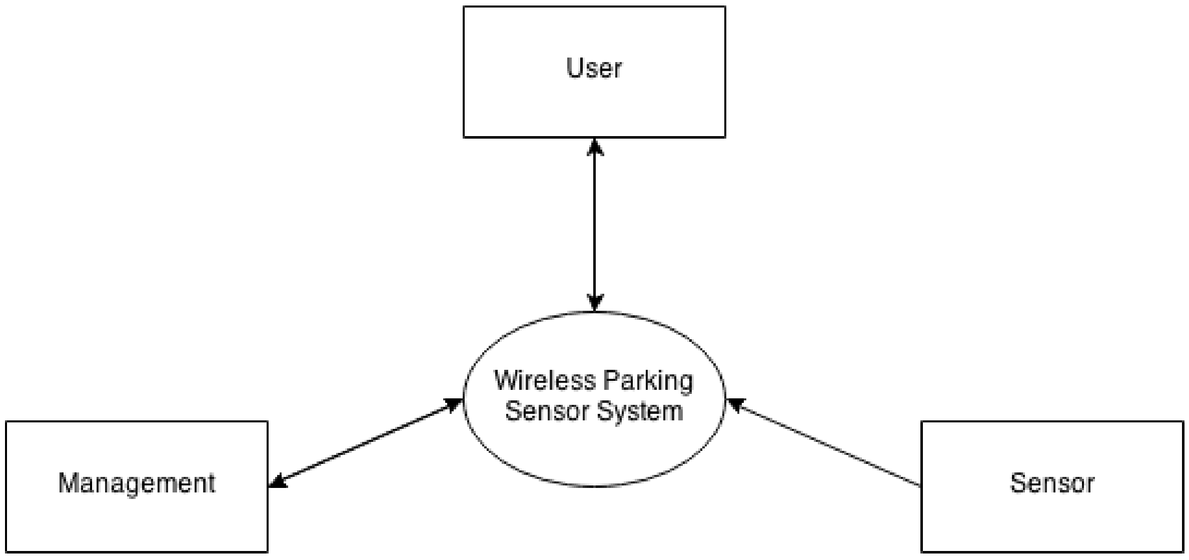


Figure 3.2.1: Context Diagram

This context diagram shows the system boundary of the software that we are going to develop, as well as the external entities that interact with the software system. Within the system boundary of our Wireless Parking Sensor System, we have three components, which are the WPSS mobile application, a server database and parking zone counter terminals.

Beyond the system boundary, we have three main entities which interact with our system, namely the user, the management and the sensor. User group is our customer. The users will request and receive information from our WPSS server through the mobile application. Management team will perform routine maintenance on the server and troubleshoot if the necessity arises. Apart from these human interacting groups, we have a sensor system which is a crucial counterpart of WPSS. The sensor will constantly supply the feedback from the environment regarding car park status to the server to ensure that our customers are being kept updated all the time.

3.3 DATA FLOW DIAGRAM

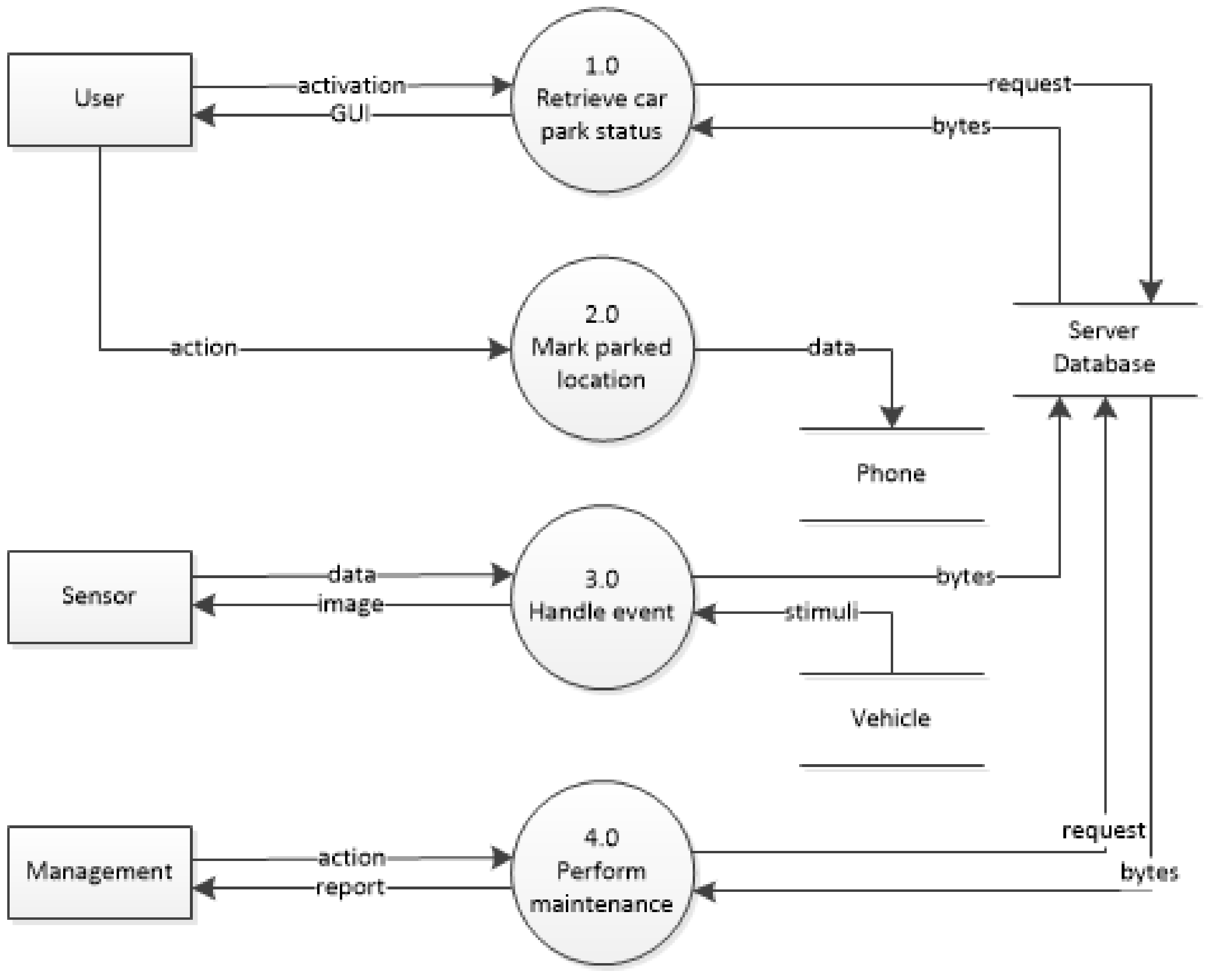


Figure 3.3.1: Data Flow Diagram Level 0

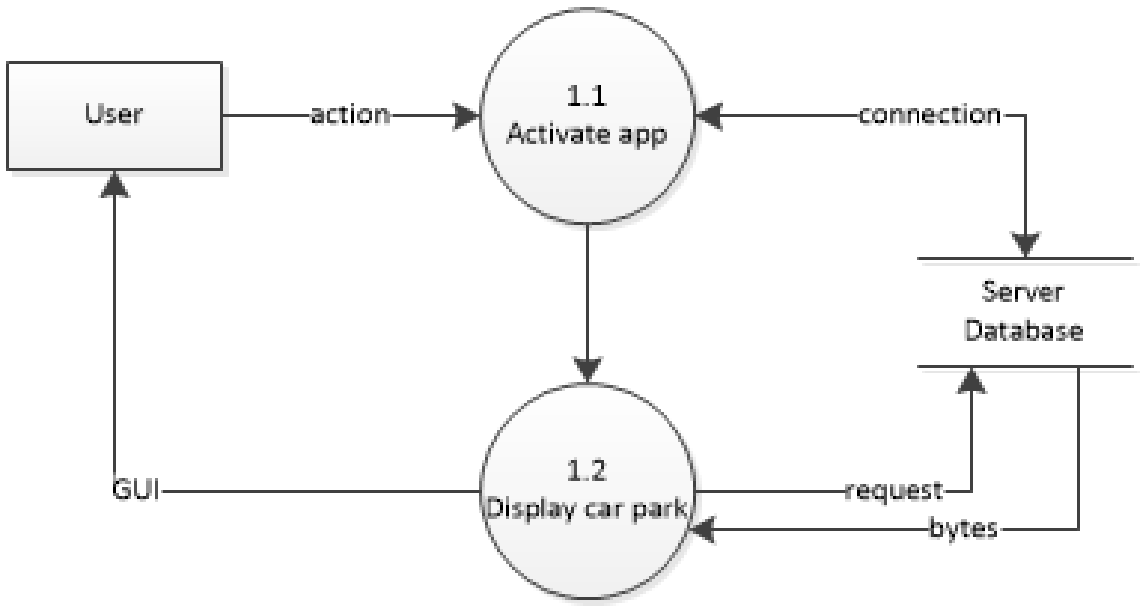


Figure 3.3.2: Data Flow Diagram Level 1 (User Application)

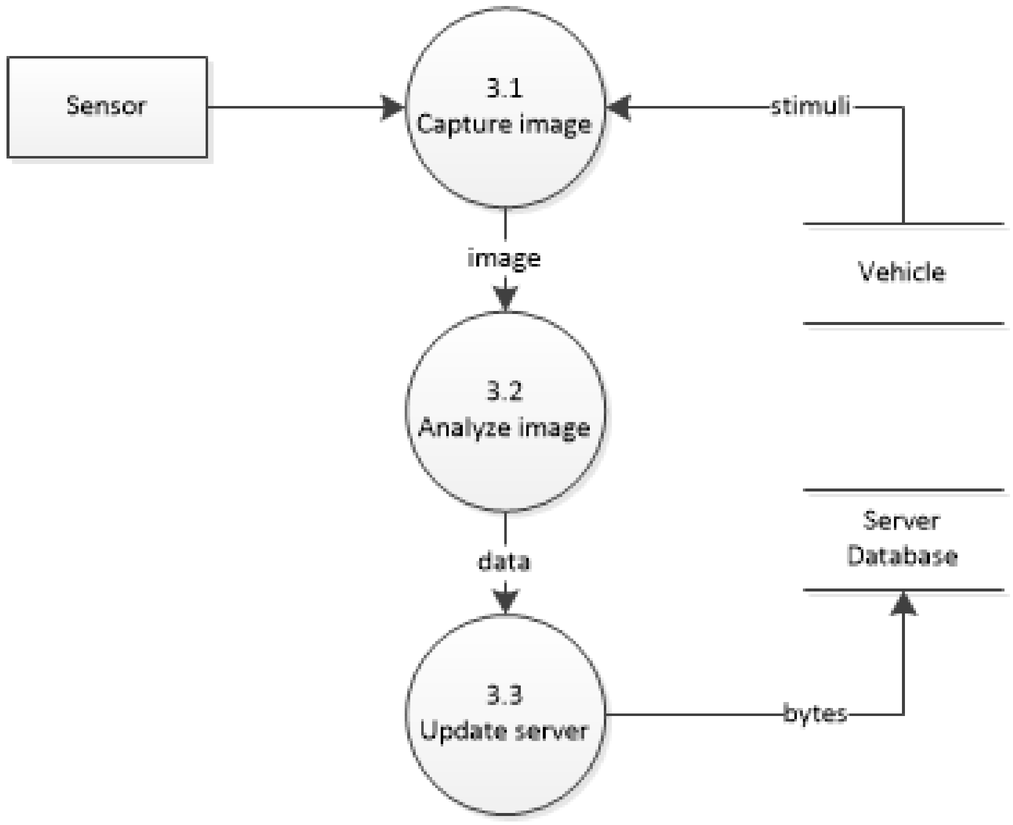


Figure 3.3.3: Data Flow Diagram Level 1 (Sensor Application)

3.4 GRAPHICAL USER INTERFACE\

1ST PROTOTYPE OF GRAPHICAL USER INTERFACE IN USER APPLICATION:

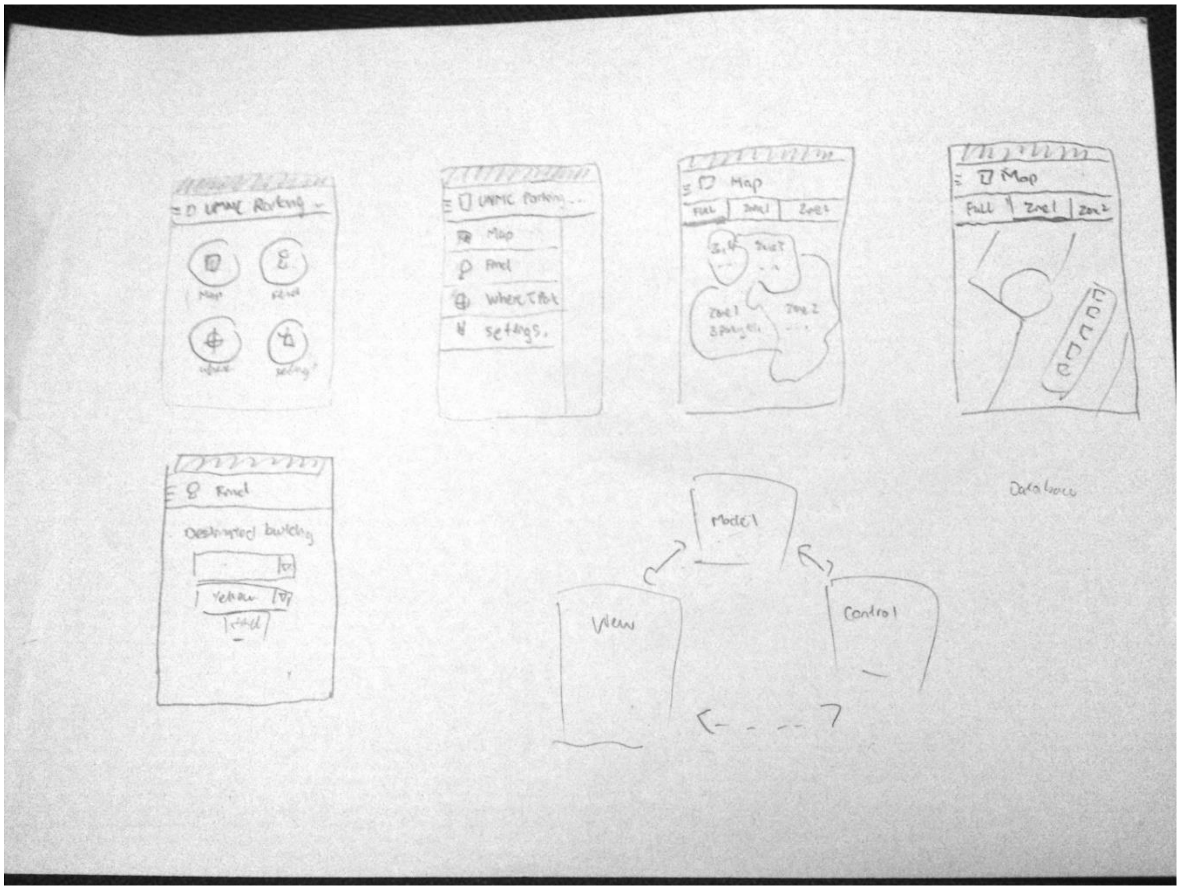


Figure 3.4.1: First Prototype of Graphical User Interface

Figure above displays layouts from meeting brainstorming session on graphical user interface design. We agreed to use the layout mostly used on Google's android applications like Google+ and Google Play Store as our guideline in designing the interface. Google provided a good documentation on designing the user interface and we decided to follow the guideline provided in our future prototypes and designs.[4]

[4] Design | Android Developers. (n.d.). Retrieved December 9, 2013, from https://developer.android.com/design/index.html

2ND PROTOTYPE OF GRAPHICAL USER INTERFACE IN USER APPLICATION:

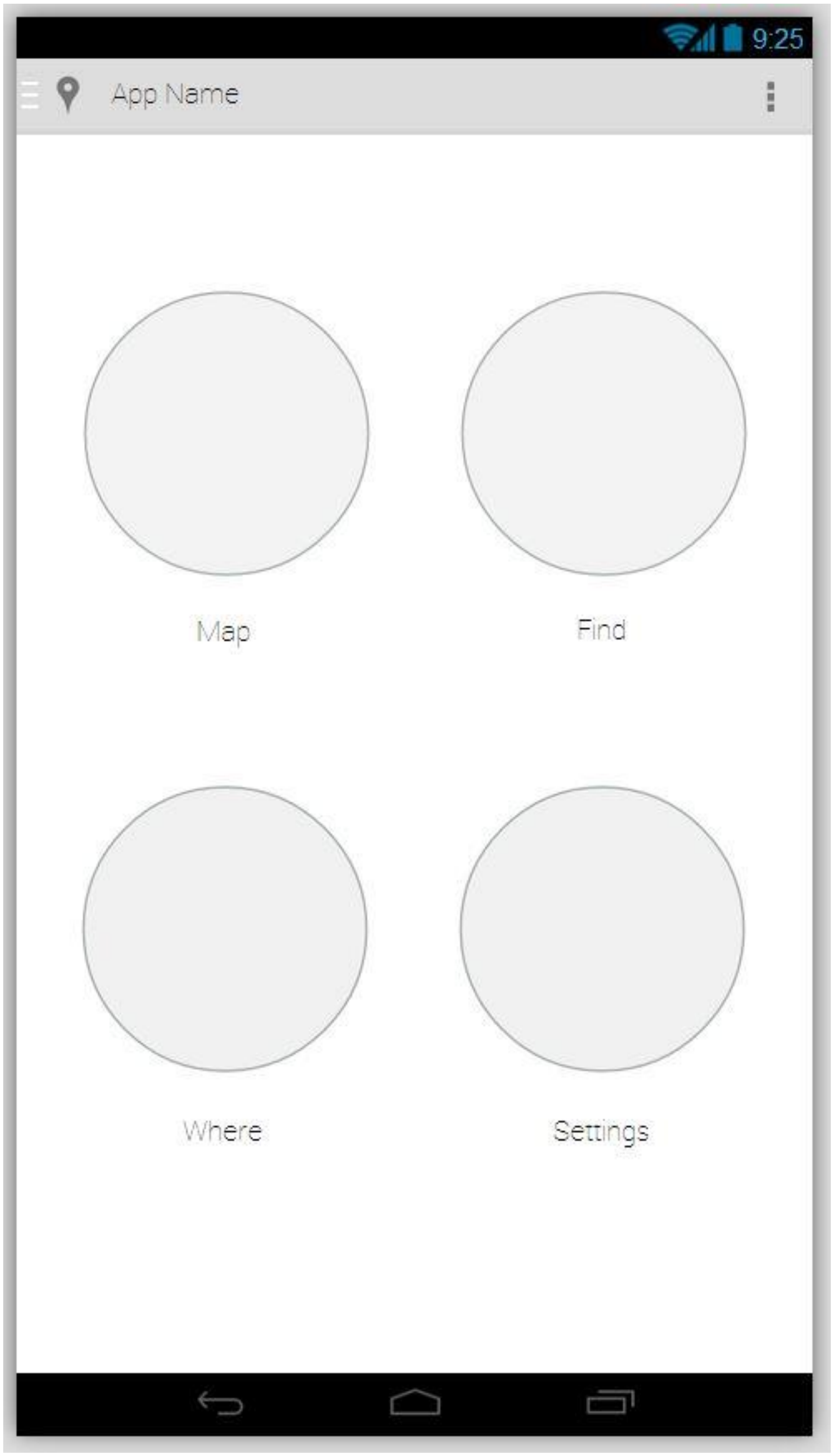


Figure 3.4.2: Main Menu

Simple 4 option plan that will leads to main features of the application. Side menu could be accessed by swiping from left edges or clicking on the icon on top left corner. Top left corner icon leads to settings page.

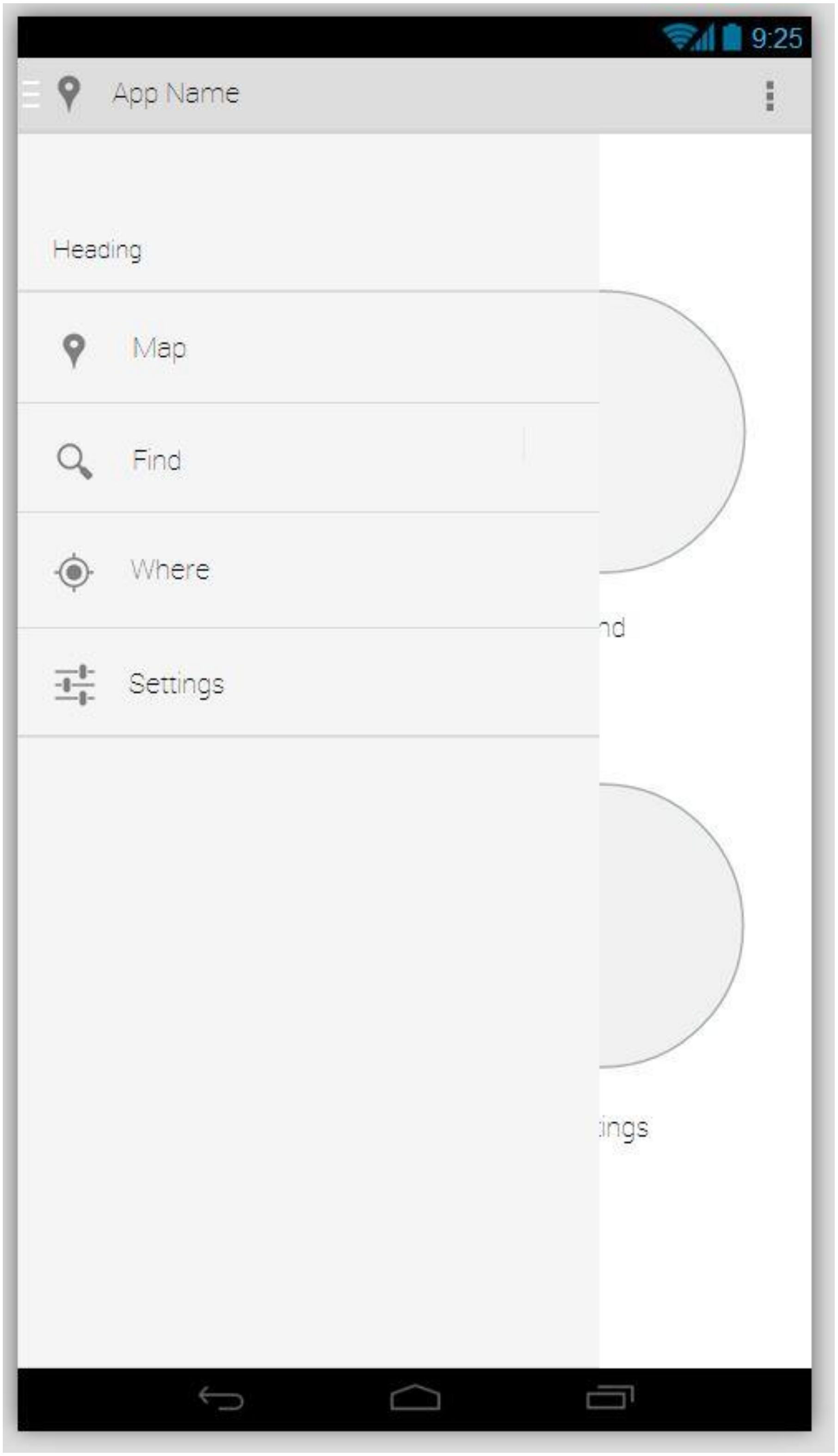


Figure 3.4.3: Side Menu

This is made up of a drop down menu which provides quick access to all the features of the application. It can be accessed by swiping from the left edge or by clicking on the top left icon.

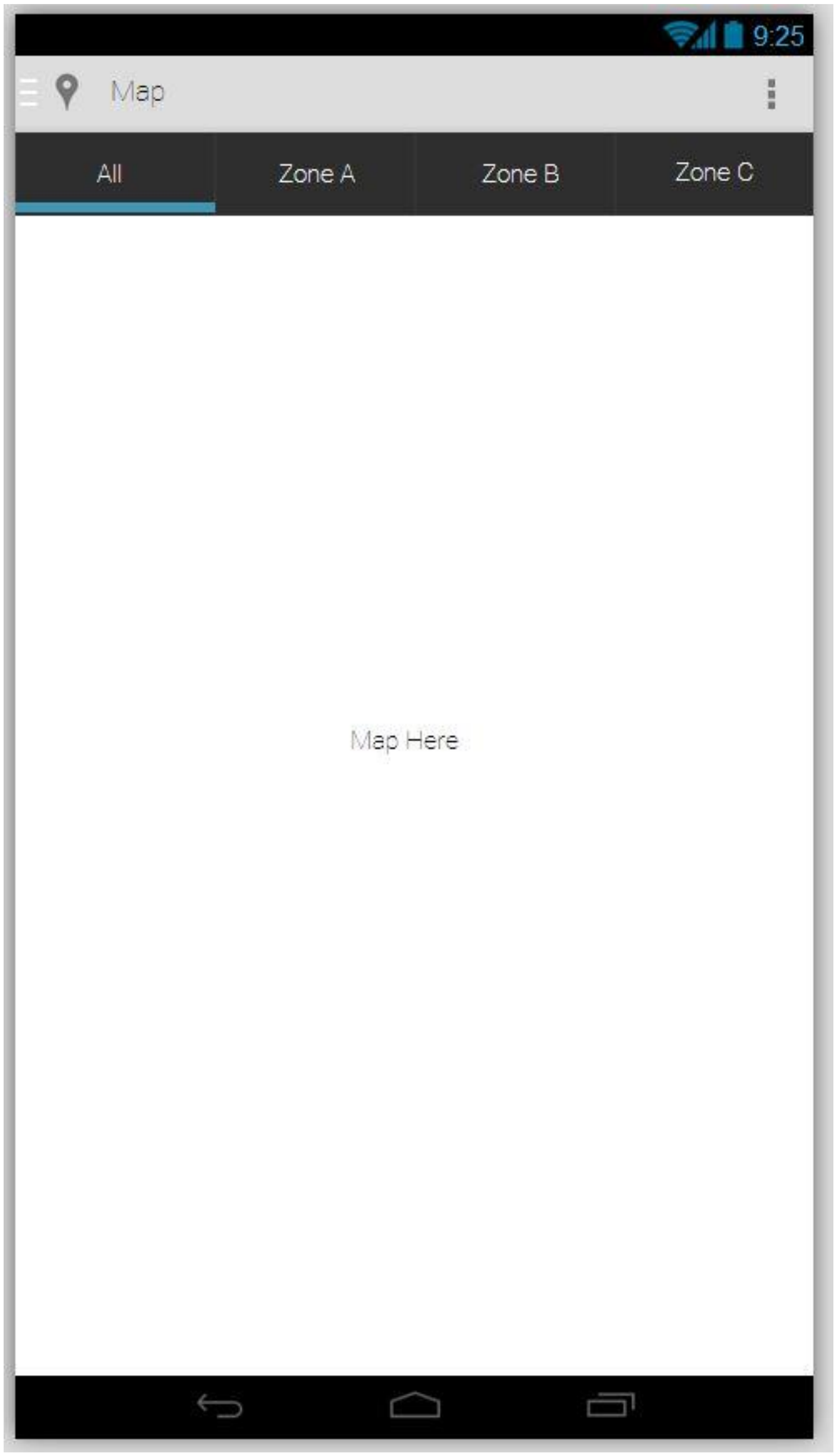


Figure 3.4.4: Map

The map will be separated into zones. The first “All” tab will display an overview map separated by zones. Each zone will display zone label and number of parking spaces left. When the user selects on one of the zones, it will lead to a detailed map of the zone itself with parking bays layouts and labels. The user could also navigate through the zones with the tab on the top. Users are also able to save their vehicle’s location by holding on a parking space for a few seconds and the system will prompt them to save it.

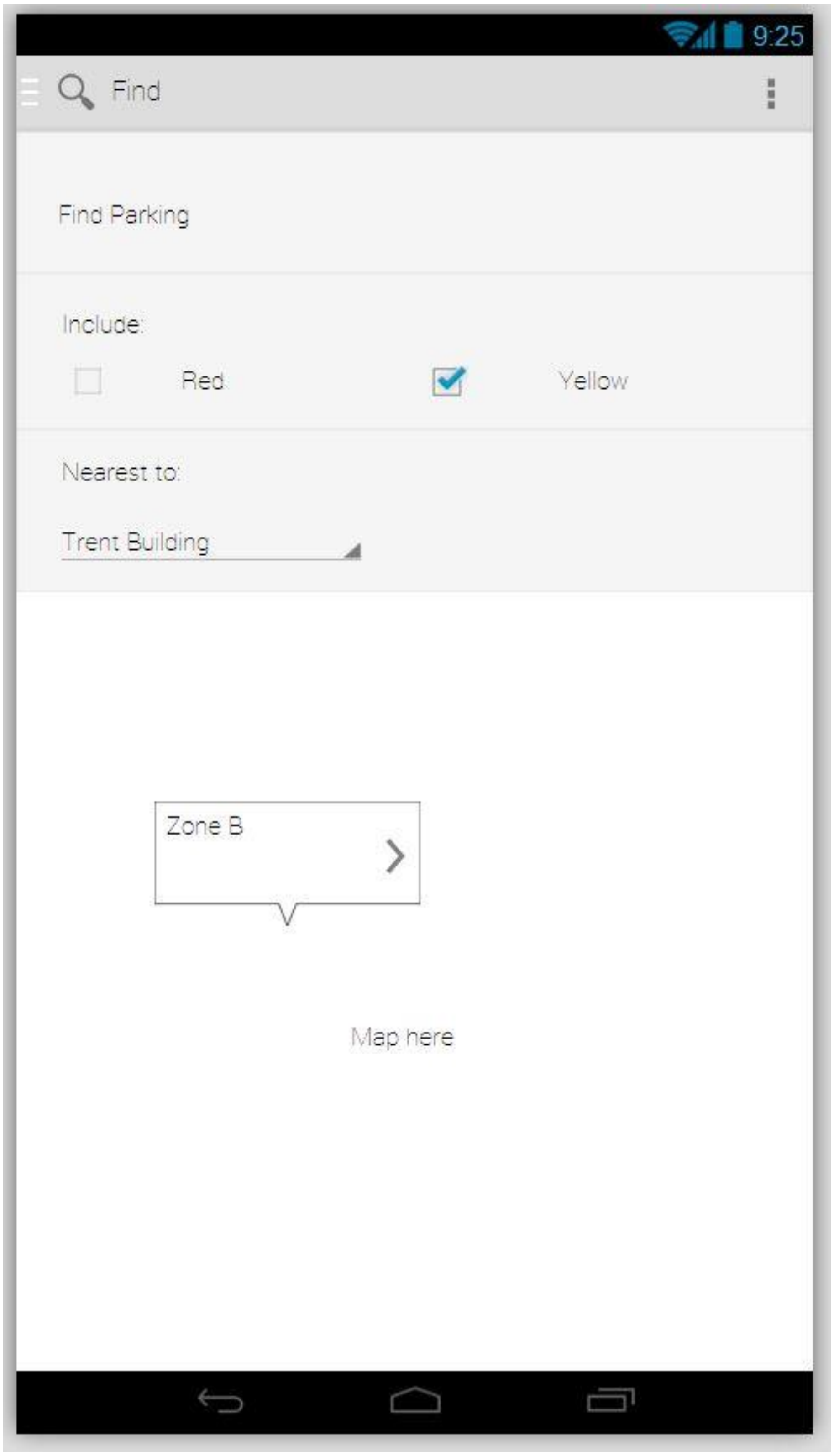


Figure 3.4.5: Find

The Find section of the application provides users with the ability to locate parking. Users are prompted to provide query arguments that are needed to search for nearest available parking. The lower bottom lays out a detailed map of the zone found to be closest and the free parking slot is labeled.

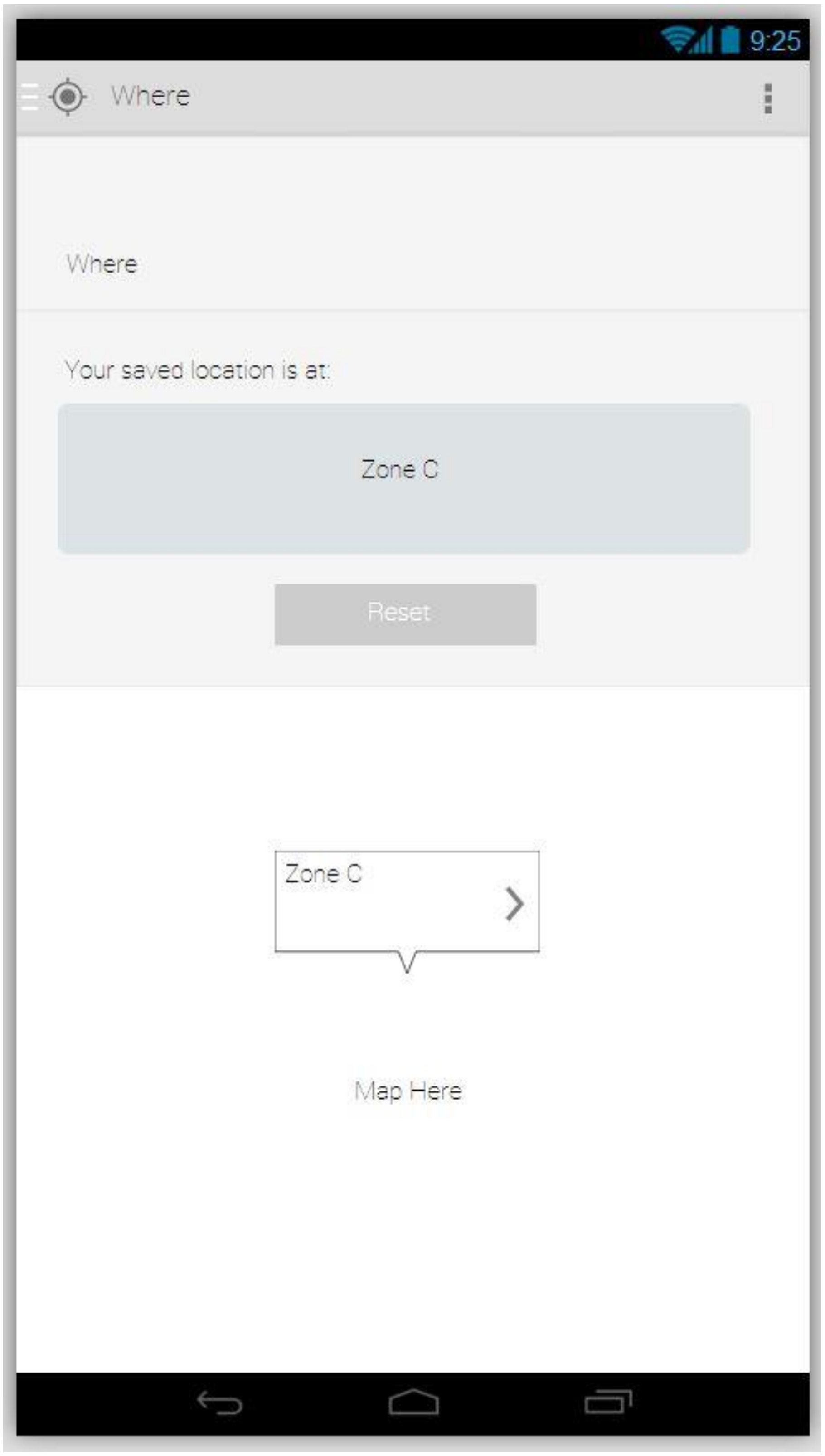


Figure 3.4.6: Where

The Where section of the application reminds the user where they have parked their vehicle. It shows the zone the vehicle is at and provides a detailed layout of the location on a map. There is also a reset button for the user to remove saved location.

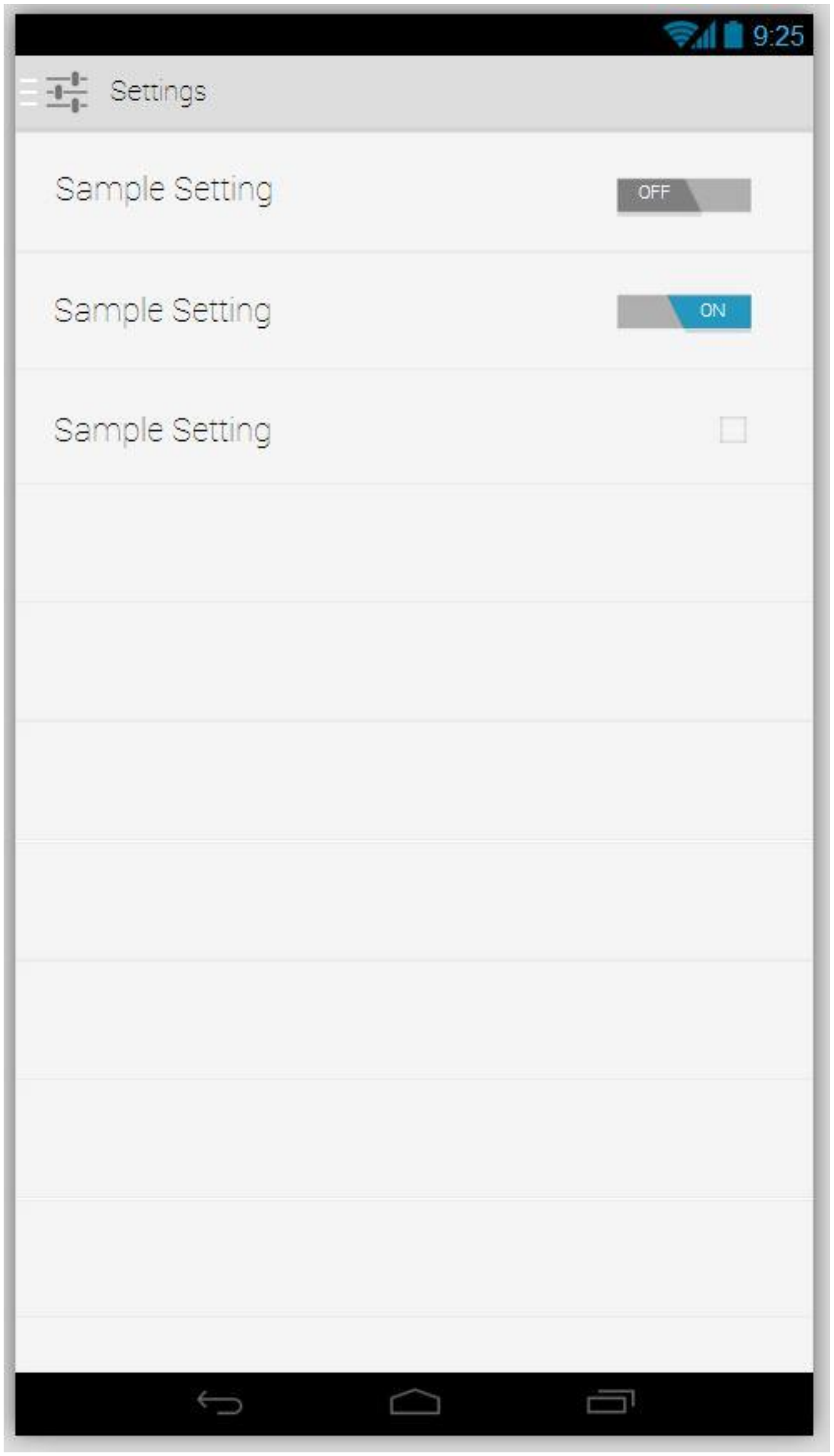


Figure 3.4.7: Settings

A simple lists of settings that soon to be added with display of sample graphical user interface toggle buttons and check box.

3.5 KEY IMPLEMENTATION

This project is carried out with the aim of creating a wireless parking system. Basically this wireless parking sensor system will consist of a server with an online database, a user Android application, a car park counter and the native camera of an Android phone, being utilized as a vehicle sensor.

The phone camera will detect whether there is a car parked at a particular parking bay and send a signal to the online database. The database will automatically update the availability of the parking bays in each car park zone. Once the users switch on the Android application on their phones, it will fetch the data of the parking space availability from the database and display it on the map layout so that the users can check on the location of the available parking bays at that moment, as well as provide various other options, such as saving a parking spot and performing searches.

All the 4 main components shall be connected via an internet connection through a given TCP/IP port, with different functions manipulating the online server and its resources. Likewise, the server shall use the same method in response.

Implementation of this shall be done in Java, as the targeted programming language for the Android User application and Sensor application, while MySQL shall be used for the back-end database.

The main reason for choosing Java is because all members of the group are familiar and proficient with it, and have experience in the language since the first year of undergraduate studies. Moreover, Java is a relatively easy language to write, compile, debug and is one of the most widely used languages. Additionally, the object-oriented nature of Java, which emphasizes on creating, manipulating and making objects work together, allows us to conveniently create and re-use modular objects and codes.

Similarly, MySQL is chosen as the platform for our database, as we have experience in using it throughout our undergraduate studies thus far. We also have knowledge of connecting it through Java applications.

For the Java Platform, we shall be using the Eclipse IDE to for coding and development. This is mainly due to the IDE being open source software, which means that it’s free. Moreover, Eclipse is often bundled with the Android SDK, and its tutorials are also mainly covered using it. Additionally, Eclipse has features such as detecting errors while coding which helps us to debug our codes before execution. The IDE also has built-in libraries which minimize the effort in importing external libraries. This along with the feature of suggesting solutions to errors makes it an invaluable tool.

As for the database, the application server platforms: WAMP and MAMP; shall be used. The reason for the both of them is that one of our members uses a MacBook. This server allows MySQL codes to be easily created and queried. It creates a local host for us to store data on our computers while performs editing and creation within a web browser, which allows for flexibility.

In regards to management, a web-based application known as Asana is what we use. It basically works by providing a common platform for all conversations within an online task manager. This was chosen, as it not only provided this on the web, but also via mobile phones, which allows for timely and effective communication and reminder for tasks.

To conduct testing and debugging of the system, Eclipse's in-built error-checker will be used while developing. After and during each major completion of a programming component, 3 major types of testing shall be done. Firstly, individual testing of developed components and modules, otherwise known as unit testing. Secondly, integration testing of each module completed, as they are combined together to form a whole system, such as the android client and the server. Finally, system testing of the entire system, as a check to see if the requirements stated earlier are met. This testing shall be done directly, by running from compilation.

For the purposes of version control, we decided to use GitHub. It is an open source version control system that manages files and directories through the use of a repository. A repository is a file server that remembers every change ever made to the files and the directories. This provides us the advantage of flexibility and the ability to recover old versions of data or examine the history of data changes.

This system allows each member to retrieve their own copy of each document and edit them concurrently. Whenever a member implements changes, the system knows and updates accordingly while at the same time handles conflicts in simultaneous updates via user input. This system is useful in controlling changes made to the software and keeping track of the changes made.

3.6 DISCUSSION OF PROBLEMS

TECHNICAL ISSUES

- Our project shall require an old android phone, to act as a sensor. This will need to be an adequately capable phone that can perform image processing, and with a minimum resolution of 5 MP.
- The parking zones shall also need to have a well-established and reliable Wi-Fi connection.
- The server and online database should be ‘always-on’ and have a stable and reliable foundation.
- We shall also need an equally updated parking layout.

MANAGEMENT ISSUES

Our group consists of few members in the team. This is as a result of the fallout of one of our group members which has resulted to an increase in workload amongst each member. This will be resolved by more collaborative efforts by each member

TIME PLAN

What have we done so far?

The beginning phase of our project has not been a smooth ride. We encountered almost a one-month lag prior to initiating the user stories gathering stage. The time lag is partly caused by the delay in the group formation during the first two weeks of the semester. Furthermore, we took two weeks to finally decided on the project description due to large selection of options available.

Fortunately, we managed to pick up our pace once we've made our decision on the project topic and we managed to deliver our Interim Report on time. For now, all of us are parallel in our goals and are clear of what to do next.

What we will do next?

Once we submitted our Interim Report, we will start our software development iterations based on the Extreme Programming methodology. As you can see from the Gantt chart, we will have three consecutive iteration phases coming up starting from 13 December 2013 and ends on 29 May 2014. Each iteration cycle spans 8 weeks and it will be a mini version of the conventional waterfall development cycle. Hence, we will have requirement refactoring, system design, coding and system testing in every cycle. We expect to have a running software which fulfils the customer requirements completely by the end of the final software implementation.

Besides, we also include the Open Day and Presentation Day milestones in our Gantt chart as they contribute significant amount of marks to our project evaluation. However, we have yet to receive the dates of these two events so that we can only set a rough estimate on their time plans with reference to previous year's academic calendars.

REFERENCE

Design | Android Developers. (n.d.). Retrieved December 9, 2013, from https://developer.android.com/design/index.html

My Car Parking - Android Apps on Google Play. (n.d.). Retrieved December 9, 2013, from https://play.google.com/store/apps/details?id=it.drd.uultimatecarfindparking&hl=en

Shopping Mall Malaysia, Sunway Pyramid. (n.d.). Retrieved December 9, 2013, from http://www.sunway.com.my/pyramid/corpInfo/mallServ_FAQ.asp

Vehicle detection by Nedap AVI. (n.d.). Retrieved December 9, 2013, from http://www.nedapavi.com/solutions/vehicle-detection.html

Data Flow Diagram (DFD). (n.d.). concept. Retrieved December 9, 2013, from http://infocenter.sybase.com/help/topic/com.sybase.infocenter.dc38088.1610/doc/html/rad1232026266129.html

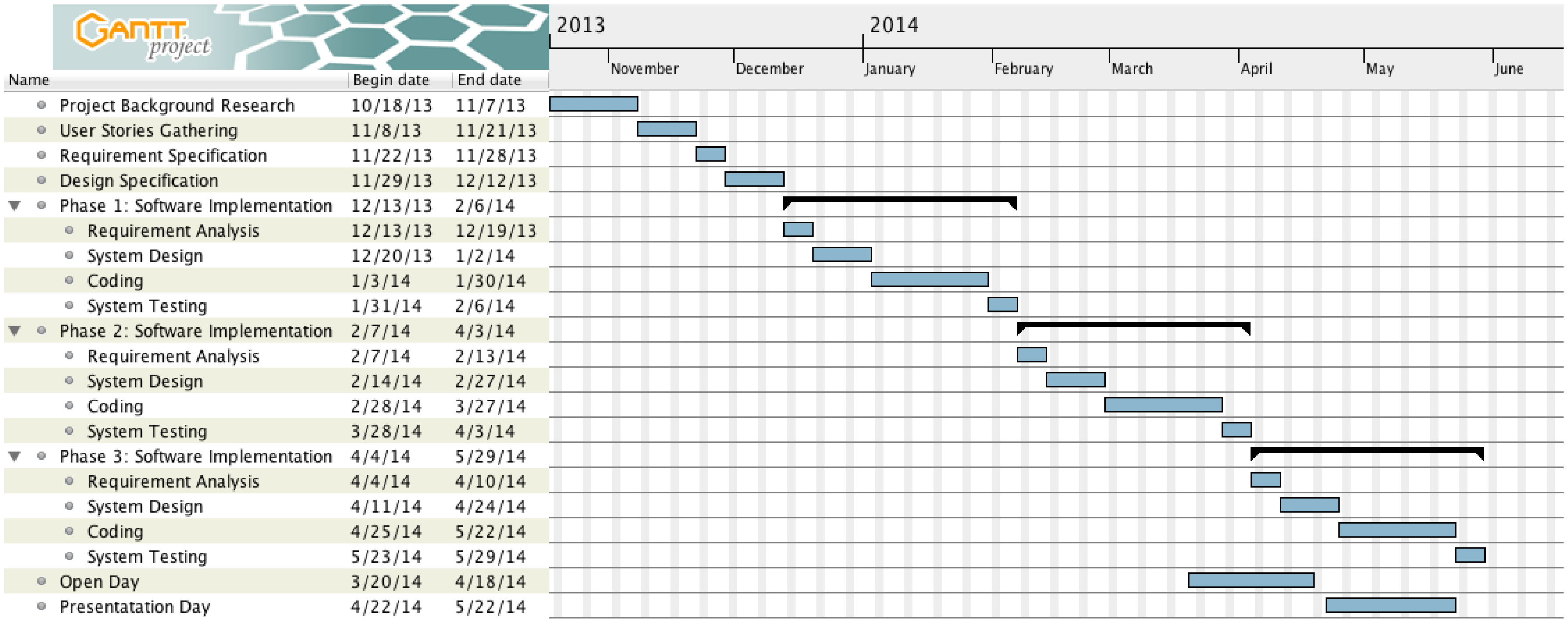
Extreme Programming. (n.d.). Retrieved December 9, 2013, from http://www.cs.usfca.edu/~parrt/course/601/lectures/xp.html

Guide to Agile Practices. (n.d.). Retrieved December 9, 2013, from http://guide.agilealliance.org/

Principles behind the Agile Manifesto. (n.d.). Retrieved December 9, 2013, from http://agilemanifesto.org/principles.html

Writing user stories — Government Service Design Manual. (n.d.). Retrieved December 9, 2013, from https://www.gov.uk/service-manual/agile/writing-user-stories.html

APPENDIX



Name	Begin date	End date
Project Background Research	10/18/13	11/7/13
User Stories Gathering	11/8/13	11/21/13
Requirement Specification	11/22/13	11/28/13
Design Specification	11/29/13	12/12/13
Phase 1: Software Implementation	12/13/13	2/6/14
Requirement Analysis	12/13/13	12/19/13
System Design	12/20/13	1/2/14
Coding	1/3/14	1/30/14
System Testing	1/31/14	2/6/14
Phase 2: Software Implementation	2/7/14	4/3/14
Requirement Analysis	2/7/14	2/13/14
System Design	2/14/14	2/27/14
Coding	2/28/14	3/27/14
System Testing	3/28/14	4/3/14
Phase 3: Software Implementation	4/4/14	5/29/14
Requirement Analysis	4/4/14	4/10/14
System Design	4/11/14	4/24/14
Coding	4/25/14	5/22/14
System Testing	5/23/14	5/29/14
Open Day	3/20/14	4/18/14
Presentatation Day	4/22/14	5/22/14

Figure